

Gaussian Mixture Models

Aayush Bajaj

July 10, 2026

Contents

1	the latent-variable formulation	1
2	responsibilities	2
3	em, derived via the elbo	2
3.1	the bound	2
3.2	m-step closed forms	3
3.3	monotonicity	3
4	singularities and regularisation	4
5	how many components?	4
6	relation to k-means	5
7	from scratch	5
8	see also	6

a single gaussian is a committed statement: one bump, symmetric, thin tails. real data is usually several stories overlaid — different regimes, different subpopulations — and a gaussian mixture says so explicitly: each point was generated by *one of k* gaussians, we just don't get told which. fitting one is the canonical latent-variable problem, and the algorithm that fits it — expectation-maximisation — is one of the great workhorses of statistics.

the 'don't get told which' is the entire difficulty, and em is the entire answer

1 the latent-variable formulation

generative story for each data point $x_i \in \mathbb{R}^d$:

- draw a component label $z_i \sim \text{Cat}(\pi_1, \dots, \pi_k)$, with $\pi_j \geq 0$, $\sum_j \pi_j = 1$;
- draw $x_i \mid z_i = j \sim \mathcal{N}(\mu_j, \Sigma_j)$.

marginalising the label gives the mixture density

$$p(x \mid \theta) = \sum_{j=1}^k \pi_j \mathcal{N}(x \mid \mu_j, \Sigma_j), \quad \theta = \{\pi_j, \mu_j, \Sigma_j\}_{j=1}^k, \quad (1)$$

which can approximate any continuous density arbitrarily well with enough compo-

nents. the log-likelihood of the data is

$$\log p(X \mid \theta) = \sum_{i=1}^n \log \sum_{j=1}^k \pi_j \mathcal{N}(x_i \mid \mu_j, \Sigma_j), \quad (2)$$

and there is the problem: a **log of a sum**. differentiate it and the components refuse to decouple; no closed-form maximiser exists (Deisenroth, Marc Peter and Faisal, A. Aldo and Ong, Cheng Soon, 2020). contrast the *complete-data* log-likelihood — if we knew every z_i , the sum inside the log would collapse to a single term and the mle would be per-component gaussian fits, a homework exercise. em manufactures exactly this simplification, in expectation.

2 responsibilities

the natural bridge between “labels known” and “labels unknown” is the posterior over labels. by bayes’ rule, the **responsibility** component j takes for point x_i is

$$r_{ij} = p(z_i = j \mid x_i, \theta) = \frac{\pi_j \mathcal{N}(x_i \mid \mu_j, \Sigma_j)}{\sum_{l=1}^k \pi_l \mathcal{N}(x_i \mid \mu_l, \Sigma_l)}, \quad (3)$$

a soft assignment: $r_{ij} \in (0, 1)$, $\sum_j r_{ij} = 1$. where **k-means** forces each point to pick a side, a mixture lets boundary points hedge.

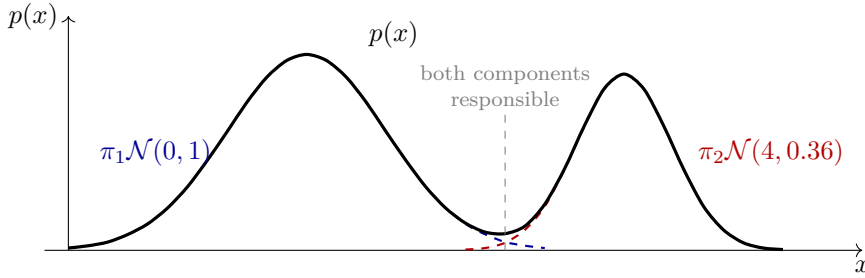


Figure 1: a two-component 1d mixture: weighted component densities $\pi_j \mathcal{N}(\mu_j, \sigma_j^2)$ (dashed) sum to the mixture density (solid). in the overlap region both components claim responsibility.

3 em, derived via the elbo

3.1 the bound

for *any* distributions $q_i(j)$ over the labels, multiply and divide inside the log-sum, then apply jensen’s inequality (log is concave, so $\log \mathbb{E}[\cdot] \geq \mathbb{E}[\log(\cdot)]$):

$$\begin{aligned} \log p(X \mid \theta) &= \sum_{i=1}^n \log \sum_{j=1}^k q_i(j) \frac{\pi_j \mathcal{N}(x_i \mid \mu_j, \Sigma_j)}{q_i(j)} \\ &\geq \sum_{i=1}^n \sum_{j=1}^k q_i(j) \log \frac{\pi_j \mathcal{N}(x_i \mid \mu_j, \Sigma_j)}{q_i(j)} =: \mathcal{L}(q, \theta), \end{aligned}$$

the **evidence lower bound** (elbo). a direct calculation shows the gap is exactly a kl divergence:

$$\log p(X \mid \theta) - \mathcal{L}(q, \theta) = \sum_{i=1}^n \text{KL}(q_i \parallel p(z_i \mid x_i, \theta)) \geq 0. \quad (4)$$

em is coordinate ascent on $\mathcal{L}$:

- **e-step**: maximise $\mathcal{L}$ over q with θ fixed. the kl gap vanishes iff $q_i = p(z_i \mid x_i, \theta)$ — set $q_i(j) = r_{ij}$, the responsibilities, making the bound *tight* at the current θ .
- **m-step**: maximise $\mathcal{L}$ over θ with q fixed. dropping the θ -free entropy term, this is maximising the **expected complete-data log-likelihood** $Q(\theta) = \sum_{i,j} r_{ij} \log [\pi_j \mathcal{N}(x_i \mid \mu_j, \Sigma_j)]$.

3.2 m-step closed forms

write $N_j = \sum_i r_{ij}$ (the effective count claimed by component j). expand the gaussian log-density in Q :

$$Q(\theta) = \sum_{i,j} r_{ij} \left[\log \pi_j - \frac{1}{2} \log |2\pi \Sigma_j| - \frac{1}{2} (x_i - \mu_j)^\top \Sigma_j^{-1} (x_i - \mu_j) \right]. \quad (5)$$

means. $\nabla_{\mu_j} Q = \Sigma_j^{-1} \sum_i r_{ij} (x_i - \mu_j) = 0$ gives a responsibility-weighted average:

$$\mu_j^{\text{new}} = \frac{1}{N_j} \sum_{i=1}^n r_{ij} x_i. \quad (6)$$

covariances. setting $\nabla_{\Sigma_j^{-1}} Q = 0$ (differentiate with respect to the precision; the two useful identities are $\partial \log |A| / \partial A = A^{-\top}$ and $\partial a^\top A a / \partial A = a a^\top$) gives a weighted scatter:

$$\Sigma_j^{\text{new}} = \frac{1}{N_j} \sum_{i=1}^n r_{ij} (x_i - \mu_j^{\text{new}})(x_i - \mu_j^{\text{new}})^\top. \quad (7)$$

weights. maximise $\sum_{i,j} r_{ij} \log \pi_j$ subject to $\sum_j \pi_j = 1$ with a lagrange multiplier: $\frac{N_j}{\pi_j} + \lambda = 0$, and enforcing the constraint fixes $\lambda = -n$, so

$$\pi_j^{\text{new}} = \frac{N_j}{n}. \quad (8)$$

every formula is the labelled-data mle with hard counts replaced by responsibility-weighted counts — the complete-data problem, solved in expectation (Deisenroth, Marc Peter and Faisal, A. Aldo and Ong, Cheng Soon, 2020).

em is not gradient descent: no learning rate, no line search, and each update is a full analytic maximisation

3.3 monotonicity

the two steps sandwich the log-likelihood upwards:

$$\log p(X \mid \theta^{t+1}) \geq \mathcal{L}(q^t, \theta^{t+1}) \geq \mathcal{L}(q^t, \theta^t) = \log p(X \mid \theta^t), \quad (9)$$

where the first inequality is “elbo is a lower bound”, the second is “the m-step maximises over θ ”, and the equality is “the e-step made the bound tight”. so **the observed log-likelihood never decreases** — printing it each iteration is both a convergence criterion and a unit test (any decrease is a bug, full stop). under mild regularity the iterates converge to a stationary point of the likelihood — typically a local maximum, with no guarantee about the global one; em inherits k-means’ dependence on initialisation, and in fact k-means is routinely used *as* the initialiser. the general em recipe — bound, tighten, maximise — works for any latent-variable model whose complete-data mle is tractable (Hastie, Trevor and Tibshirani, Robert and Friedman, Jerome, 2009).¹

4 singularities and regularisation

the gmm likelihood has a dirty secret: its supremum is $+\infty$. park one component exactly on a single data point, $\mu_j = x_i$, and shrink its covariance: the density $\mathcal{N}(x_i | x_i, \sigma^2 I) \propto \sigma^{-d} \rightarrow \infty$ as $\sigma \rightarrow 0$, while every other point stays covered by the remaining components. the “maximum likelihood solution” is a spike of zero width and infinite height — a component that has memorised one observation. defences, in ascending order of principle:

you will meet this in practice as a covariance collapsing toward zero and the log-likelihood soaring — success-shaped failure

- **variance floor:** clamp $\Sigma_j \succeq \varepsilon I$ (the + `eps` in the code below).
- **map estimation:** put an inverse-wishart prior on each Σ_j ; the m-step gains an additive prior scatter and can no longer collapse.
- **detect and restart:** if a component’s determinant crashes, re-seed it and continue.

a proper bayesian treatment integrates θ out and has no singularity at all — the spike has vanishing posterior mass.

5 how many components?

more components never lower the maximised likelihood, so k must be chosen by penalised fit. the **bayesian information criterion**,

$$\text{bic}(k) = -2 \log \hat{L}_k + p_k \log n, \quad p_k = \underbrace{(k-1)}_{\text{weights}} + \underbrace{k d}_{\text{means}} + \underbrace{k \frac{d(d+1)}{2}}_{\text{covariances}}, \quad (10)$$

trades likelihood against parameter count with a sample-size-aware penalty; pick the k minimising it (Hastie, Trevor and Tibshirani, Robert and Friedman, Jerome, 2009). the covariance term dominates p_k in even modest dimension — full-covariance mixtures are parameter-hungry, which is why constrained families (diagonal, tied,

¹em is due to dempster, laird & rubin (1977), *maximum likelihood from incomplete data via the em algorithm*, jrss-b 39(1) — one of the most-cited statistics papers ever written. the elbo view presented here is the modern one; it is also the launchpad for variational inference, where the e-step’s exact posterior is itself replaced by an optimised approximation.

references

Deisenroth, Marc Peter and Faisal, A. Aldo and Ong, Cheng Soon (2020). *Mathematics for Machine Learning*, Cambridge University Press. Hastie, Trevor and Tibshirani, Robert and Friedman, Jerome (2009). *The Elements of Statistical Learning*, Springer.

spherical) exist. cross-validated held-out log-likelihood is the honest alternative when you can afford it. and remember the quiet assumption underneath: bic selects the best *mixture-of-gaussians* description, not the “true number of clusters” — a skewed unimodal blob is happily described by two overlapping gaussians.

6 relation to k-means

fix all weights equal and all covariances to ϵI , and let $\epsilon \rightarrow 0$: responsibilities harden into nearest-centroid indicators, the e-step becomes lloyd’s assignment step, and the weighted-mean m-step becomes the plain centroid update — **k-means** is degenerate em. the practical reading: a gmm buys you soft boundary points, per-cluster shape (anisotropic covariances), per-cluster mass, and an actual density you can evaluate — at the price of $O(kd^2)$ parameters, slower iterations, and livelier failure modes.

7 from scratch

em on 1d data drawn from $0.65\mathcal{N}(0, 1) + 0.35\mathcal{N}(4, 0.6^2)$ — the mixture in the figure — from a deliberately bad initialisation (both means near zero):

```

1  import numpy as np
2
3  rng = np.random.default_rng(1)
4
5  # 1d data: two components, unequal weights and spreads
6  n = 400
7  z_true = rng.random(n) < 0.35 # p(component 2) = 0.35
8  x = np.where(z_true, rng.normal(4.0, 0.6, n), rng.normal(0.0, 1.0, n))
9
10 def normal_pdf(x, mu, var):
11     return np.exp(-(x - mu)**2 / (2 * var)) / np.sqrt(2 * np.pi * var)
12
13 def em(x, iters=60, eps=1e-6):
14     # deliberately poor initialisation
15     pi, mu, var = 0.5, np.array([-1.0, 1.0]), np.array([1.0, 1.0])
16     ll_old = -np.inf
17     for t in range(iters):
18         # e-step: responsibilities
19         p1 = (1 - pi) * normal_pdf(x, mu[0], var[0])
20         p2 = pi * normal_pdf(x, mu[1], var[1])
21         r = p2 / (p1 + p2) # resp. of component 2
22         # m-step: closed forms
23         n2 = r.sum()
24         pi = n2 / len(x)
25         mu = np.array([(1 - r) * x.sum() / (len(x) - n2),
26                       (r * x.sum()) / n2])
27         var = np.array([(1 - r) * (x - mu[0])**2.sum() / (len(x) - n2) + eps,
28                       (r * (x - mu[1])**2.sum() / n2 + eps)])
29         ll = np.log(p1 + p2).sum() # log-lik at *previous*
30         ↪ params
31         if t % 10 == 0 or t == iters - 1:
32             print(f"iter {t:>2}: log-lik {ll:>9.3f} pi2={pi:.3f} "
33                   f"mu=({mu[0]:.3f},{mu[1]:.3f}) "
34                   f"sd=({np.sqrt(var[0]):.3f},{np.sqrt(var[1]):.3f})")
35         if abs(ll - ll_old) < 1e-8:
36             print(f"converged at iter {t}")
37             break
38         ll_old = ll
39     return pi, mu, var
40
41 pi, mu, var = em(x)
42 print(f"\nrecovered: pi2={pi:.3f}, mu=({mu[0]:.3f}, {mu[1]:.3f}), "
```

```

42     f"sd={np.sqrt(var[0]):.3f}, {np.sqrt(var[1]):.3f})")
43 print(f"truth:      pi2=0.350, mu=(0.000, 4.000), sd=(1.000, 0.600)")
44 print(f"empirical: pi2={z_true.mean():.3f}, "
45       f"mu={({x[~z_true].mean():.3f}, {x[z_true].mean():.3f}), "
46       f"sd={({x[~z_true].std():.3f}, {x[z_true].std():.3f})")

```

```

iter 0: log-lik -1312.407  pi2=0.674  mu=(-0.623, 2.424)  sd=(0.800,1.850)
iter 10: log-lik -790.622  pi2=0.424  mu=(-0.196, 3.646)  sd=(0.883,1.088)
iter 20: log-lik -752.064  pi2=0.356  mu=(-0.008, 4.042)  sd=(1.021,0.594)
converged at iter 27

```

```

recovered: pi2=0.355, mu=(-0.008, 4.042), sd=(1.021, 0.594)
truth:     pi2=0.350, mu=(0.000, 4.000), sd=(1.000, 0.600)
empirical: pi2=0.357, mu=(-0.019, 4.039), sd=(1.005, 0.592)

```

the log-likelihood climbs monotonically ($-1312 \rightarrow -791 \rightarrow -752$) exactly as the sandwich argument promised, and from an initialisation where both means sat inside the left blob, em still pries the components apart and lands within a hair of the *empirical* per-component statistics — which, not the generating parameters, are the best any estimator could do on this sample. note the **eps** variance floor riding along in the m-step: on this dataset it never binds, but it is the difference between a demo and a fit that survives an unlucky seed.

8 see also

- **k-means clustering** — this model with the soft parts frozen hard
- **principal component analysis** — linear latent structure; gmms give multimodal latent structure
- **naive bayes** — class-conditional gaussians *with* the labels observed: the complete-data problem em emulates
- **bias-variance decomposition** — the trade bic is arbitrating, in its model-selection costume